Network-wide monitoring through self-configuring adaptive system

Imed Lassoued, Amir Krifa, Chadi Barakat, Konstantin Avrachenkov

Network-wide monitoring through self-configuring adaptive system

Imed Lassoued, Amir Krifa, Chadi Barakat, Konstantin Avratchenkov
Planete + Maestro Project-Team, INRIA, France
{First Name.Last Name}@inria.fr

***Abstract*—The remarkable growth of the Internet infrastructure and the increasing heterogeneity of applications and users' behavior make more complex the manageability and monitoring of ISP networks and raises the cost of any new deployment. The main consequence of this trend is an inherent disagreement between existing monitoring solutions and the increasing needs of management applications. In this context, we present the design of an adaptive centralized architecture that provides visibility over the entire network through a network-wide cognitive monitoring system. Practically, given a measurement task and a constraint on the volume of collected information, the proposed architecture drives the sampling rates on the interfaces of network routers to achieve the maximum possible accuracy, while adapting itself to any change in network traffic conditions. We illustrate our work with an accounting application whose purpose is to estimate the volume of aggregate flows across a backbone transit network. The paper provides a global study of the functioning of the proposed system and the impact of the different parameters on its behavior. The performance of our system is validated in typical scenarios over an experimental platform we developed for the purpose of the study.**

I. Introduction

Traffic measurement and analysis are two important management activities for network operators that allow them to determine the composition of network traffic, to understand the behavior of users, and to monitor the performance of the deployed network infrastructure. This can serve several purposes such as traffic engineering, network resource provisioning and management, accounting and anomaly detection.

Actually, there is an increasing interest in efficient passive measurement solutions that scale with the network traffic while providing the maximum possible information. Existing solutions, e.g. [1], often balance between number of monitored routers and volume of captured and exported data to meet application requirements while limiting the overhead on both routers and network links. Most of them [2], [3] treat routers independently of each other and function by static configuration, while few [4], [5] correlate the information collected from different routers to further increase the accuracy of measurements. Despite this wealth of solutions, there is a lack of intelligent monitoring architectures that self-configure themselves based on the monitoring application needs and network conditions, and that lead to the best balance between accuracy and overhead.

This research work is funded by the European Commission through the ECODE project (INFSO-ICT-223936) of the European Seventh Framework Programme (FP7).

Currently, NetFlow [6] constitutes the most popular tool for network monitoring at the flow level. While the deployment of NetFlow lowers the cost of processing and storage resources, it clearly reduces the accuracy of measurements and entails a loss of information. This is unfortunately unavoidable given the actual disagreement between the increasing speed of links and the router resource constraints. The main consequence of this trend is a fundamental disagreement between the existing monitoring capabilities and the increasing requirements of network management applications in terms of accurate measurements. The solution to this discrepancy certainly has to pass by multiplying the monitoring points inside the network, coupling their observations and finding the best network-wide configuration as a function of monitoring needs.

We introduce in this paper a network-wide cognitive monitoring system that benefits from advances in machine learning techniques and flow-level monitoring tools. The system starts from the NetFlow monitoring capabilities deployed in routers, tunes them in an adaptive and optimal way, and combines their results to answer to the best the monitoring application needs. Our system is centralized and proceeds in optimizing the configuration in small steps based on dynamics inspired from the one used by TCP for the adjustment of its congestion window and using the gradient Projection Method (GPM) to identify the monitors that we should reconfigure until the optimal configuration is reached. The main configuration parameter that we tune is the *packet sampling rate*, which decides the amount of traffic collected by each monitor. Conducting real experiments on a platform we developed for the purpose of the study, we show how our system can drive its own deployment and configuration to the optimal by diagnosing the reported network traffic, learning about the status of flows and the accuracy of estimators, and taking the best adjustment decisions on the sampling rates in monitors.

This paper is organized as follows. In Section II, we present some works related to traffic measurement and monitoring systems. Then, we present in Section III the proposed system architecture as well as the algorithms that drive it. In Section IV we give a detailed explanation of an accounting application as an example of monitoring task to realize. Our experimental testbed is presented in Section V. Performance evaluation results and parameters importance analysis are discussed in Section VI. Finally, conclusions and future work are given in Section VII.

II. Related work

The interest in passive monitoring for the understanding and diagnosis of core IP networks has grown at an astonishing rate. Recently, numerous works have focused on the design of new monitoring techniques and on the analysis of the collected results. Currently, NetFlow [6], [2] is the most widely deployed measurement solution by ISPs. However, this solution still presents some shortcomings, namely the problem of configuring sampling rates according to network conditions and the requirements of monitoring applications. Another problem comes from the low values to which the sampling rate is set in practice (between 0.01 and 0.001) to be able to cope with the increasing trend in line speed.

Some recent proposals provide a network-wide monitoring infrastructure that distributes the work between the different monitors. For instance, the authors of [7] argue that performance limits can be addressed by reducing the sampling rates in the different monitors while accuracy can be improved by combining the different measurements of each flow. The authors in [4] present a system-wide approach that uses a hash-based flow selection to eliminate duplicate measurements in the network and a framework for distributing responsibilities across routers. Our architecture adds another dimension to the problem, that of packet sampling, to allow further reduction of the load, while working towards the optimization of sampling rates at the network level to reduce the error during the traffic estimation phase. This architecture meets the one in [5] in which the interest of network-wide optimization of sampling rates is shown in a simple simulation scenario, that of estimation of the traffic volume between Autonomous Systems (ASes). We generalize these ideas to NetFlow and traffic accounting, propose an online optimization algorithm, bridge the gap between theory and implementation, and validate the ensemble over a real network emulation platform [8].

III. System architecture

The main goal is to build a network-wide system that, given a measurement task and a target overhead value $\mathcal{TO}$ (the $\mathcal{TO}$ is defined as the desired rate of reported flow records), adaptively adjusts its configuration according to network conditions and measurement accuracy. We adopt a centralized approach that relies on a central unit or collector and we extend the local existing monitoring tools (NetFlow) with a network-wide cognitive system able to:

- Investigate the measurements collected from the different routers (local views) and then construct a global view of the traffic and the network state.
- Drive its own deployment by automatically reconfiguring the different monitors. The automation of the control of sampling rates is achieved by learning experiences from the accuracy of the collected data and the resulting overhead $\mathcal{O}$.

This central collector is composed of two main modules:

Global Network Traffic Inference Engine: Given a measurement task T to realize, this inference engine investigates the local measurements made by the different routers to obtain a more reliable global view. The engine takes as inputs the sampling rate vector of network routers as well as the local estimations of T, $(\hat{T}_k)_{k=1..M}$, calculated from the reports sent to the collector by the different routers. M is the number of monitor or equivalently, router interfaces in the network. The inference engine then tries to combine the local estimators and to derive a better global estimation of T. This combination is motivated by the need to minimize the variance of the global estimation error. To this end, we construct the global estimator of the task T as a weighted sum of the different local estimators. This weighted summation of local independent estimators is known to be the best linear combination in terms of mean square error [7],

$$\hat{T} = \sum_k \lambda_k \hat{T}_k \quad \text{with} \quad \lambda_k = \frac{\frac{1}{Var(\hat{T}_k)}}{\sum_l \frac{1}{Var(\hat{T}_l)}}. \tag{1}$$

Note that the summation is only done over monitors that are supposed to see the traffic of interest. The weights are inversely proportional to the local estimator errors, which in their turn are inversely proportional to the configured sampling rates. Thus, local estimates with smaller error variance have a larger impact on the global estimator than those with larger errors.

Network Reconfiguration Engine: This engine is motivated by the need to coordinate responsibilities across the different monitors in order to increase global accuracy while avoiding unnecessary measurements. To do so, we proceed by an adaptive centralized control of sampling rates based on the estimation of the measurement error and the reporting overhead (as shown in Algorithm 1). We resort to a dynamics inspired from the one used by TCP for the adjustment of its congestion window. Starting from an initial sampling rate vector $\mathcal{P}_{init}$, the Network Reconfiguration Engine is fed with the estimation of task T and the variance of this estimation ($Var(\hat{T})$), as well as the resulting overhead ($\mathcal{O}$), i.e. rate at which flow records arrive. If the $\mathcal{O}$ is less than the target $\mathcal{TO}$, the system keeps increasing periodically (each t) the sampling rates of the different monitors. Once $\mathcal{TO}$ is reached, the system triggers a decrease in the sampling rates of the least significant monitors. In this way the system strives to keep the reporting overhead at $\mathcal{TO}$ flow records per second and fully profits from the available resources. Note that setting the sampling rate to a very low value in a router (SR_{min}) is equivalent to turning it off for the purpose of monitoring while we don't let the sampling rate exceeds some maximum value (SR_{max}). to respect local router constraints.

To increase or to decrease sampling rates, we use increments in the logarithmic scale in order to give more flexibility to our system and to get a fast scan of the interval $[0, 1]$. For reconfiguring the sampling rate of, let's say monitor k, we set $log(p_k)$ to $log(p_k) \pm \delta$. This gives in the normal scale $p_k = p_k(\gamma)^{\pm 1}$ where $\gamma = exp(\delta)$. In our experiments, we measure $\mathcal{O}$ and we set the value of γ at $\min\{1+\sigma|\frac{\mathcal{TO}-\mathcal{O}}{\mathcal{TO}}|, 3\}$, so that this value varies between 1 and 3. $\mathcal{O}$ is the number of flow records received since the last update, divided by the time

since this last update. It is immediately noticed that the value of γ depends on the value of $\mathcal{O}$: small adjustments when $\mathcal{O}$ converges to $\mathcal{TO}$ and large adjustments when $\mathcal{O}$ deviates from $\mathcal{TO}$. σ is a constant parameter of the control that represents a balance between convergence speed and stability.

The least significant monitors are identified using the Gradient Projection Method (GPM). From the perspective of the task T, the least significant monitors are the ones providing the least increase in the variance of the estimator of T, i.e. $Var(\hat{T})$, when the logarithmic of their sampling rates are decreased by step δ. To be identified, one has first to write analytically the expression of $Var(\hat{T})$ as a function of the sampling rates in routers of interest, then calculate the utility function of the different monitors by differentiating this expression with respect to $log(p_k)$, where p_k is the sampling rate of the monitor k: $U_k = |\partial Var(\hat{T})/\partial log(p_k)|$, $k = 1 \dots M$. Given the current configuration of sampling rates, we choose the least significant monitors as being those having utility function values less than the average of the utility functions values over all the monitors.

Data: The global estimation $\hat{T}$ with its estimation errors $Var(\hat{T})$, and the sampling rate vector $\mathcal{P}$
Result: The new sampling rate vector $\mathcal{P}$
begin
  Initialize the sampling rate vector at $\mathcal{P}_{init}$;
  $\mathcal{P} \leftarrow \mathcal{P}_{init}$;
  while $True$ **do**
    /* If $\mathcal{O}$ exceeds $\mathcal{TO}$, the system triggers a decrease in the sampling rates of the least significant monitors */ **if** $\mathcal{O}$ *exceeds* $\mathcal{TO}$ **then**
      calculate $\gamma = \min\{1 + \sigma|\frac{O-TO}{TO}|, 3\}$;
      foreach $p_k \in \mathcal{P}$ **do**
        calculate $U_k = |\partial Var(\hat{T})/\partial log(p_k)|$;
      end
      calculate $AvgUtility = Avg_{p_k \in \mathcal{P}} U_k$;
      foreach $p_k \in \mathcal{P}$ **do**
        if $U_k < AvgUtility$ **then**
          $p_k \leftarrow \max\{\frac{p_k}{\gamma}, SR_{min}\}$;
        end
      end
      return $\{\mathcal{P}\}$ rst(t, $\mathcal{O}$) ;
    end
    /* If t expires, we increase the sampling rate of the different monitors.*/ **if** t *expires* **then**
      calculate $\gamma = \min\{1 + \sigma|\frac{O-TO}{TO}|, 3\}$;
      foreach $p_k \in \mathcal{P}$ **do**
        $p_k \leftarrow \min\{\gamma p_k, SR_{max}\}$;
      end
      return $\{\mathcal{P}\}$ rst(t, $\mathcal{O}$) ;
    end
  end
end

Algorithm 1: The adaptive centralized control algorithm

IV. Case study: Traffic accounting

We explain in this section using a concrete example how the central cognitive engine, based on the collected measurements, can decide on the way to tune the sampling rates over the network. We consider for this purpose an accounting application: the estimation of the volume of some chosen network flows.

A. Definitions

Consider N traffic aggregate flows whose volumes in packets are labeled F_1, F_2, ..., F_N. Denote by $\hat{F}_1$, $\hat{F}_2$, ..., $\hat{F}_N$ the corresponding global estimators. Let $\mathcal{P}$ be the vector of sampling rates in the different monitors of the network (a monitor is equivalent to a router interface). The target of the system is to find the vector $\mathcal{P}$ that minimizes the sum of normalized estimation errors $\sum_i Var(\hat{F}_i)/F_i^2$.

Each aggregate flow F_i is formed of a set of 5-tuple flows whose volumes are denoted by S_{ji}. Again, denote by $\hat{S}_{ji}$ the best global estimator for the size of each of these 5-tuple flows. One can then transform the optimization problem into minimizing the sum of the normalized estimation errors of the sizes of the 5-tuple flows $\sum_i \sum_j Var(\hat{S}_{ji})/F_i^2$.

As long as there are available resources, the system periodically increases all sampling rates to improve measurement accuracies. Once the $\mathcal{TO}$ value is reached, the system triggers a decrease in the sampling rate of the least significant monitors. This continues until the overhead is again below the $\mathcal{TO}$. The least significant monitors are the ones having the smallest absolute values for the following partial derivation sum:

$$\sum_i \sum_j \frac{\partial Var(\hat{S}_{ji})}{\partial log(p_k)} \cdot \frac{1}{F_i^2}. \tag{2}$$

In the following we show how such estimators for the 5-tuple flow sizes are formed and how the partial derivatives of their variances are obtained. For the F_i themselves, which are unknown, we simply substitute them by their estimations, i.e. $\hat{F}_i = \sum_j \hat{S}_{ji}$. Note that we consider the volumes of flows as measured in packets. The passage to bytes can be made by multiplying the size in packets by the average packet size, which we suppose true for large flows.

B. Local flow size estimation

Consider a 5-tuple flow S_{ji} crossing monitor k whose sampling rate is p_k. Let s_{kji} be the number of packets sampled from this 5-tuple flow in monitor k (this number could be zero). With this information, one can derive a first estimation for the flow size of the flow. The estimator that maximizes the likelihood is known to be [3]: $\hat{S}_{kji} = s_{kji}/p_k$. Under independent sampling of packets with probability p_k, the number of packets s_{kji} sampled from an original 5-tuple flow S_{ji} follows a binomial distribution whose variance is well known to be equal to $S_{ji}.p_k.(1-p_k)$. It follows that this local estimator for the size of a 5-tuple flow has a variance equal to $Var(\hat{S}_{kji}) = S_{ji}.(1-p_k)/p_k$.

C. Combining measurements

The information on a 5-tuple flow comes from all monitors along its path. Though, some monitors may not sample any of the packets of the flow, either because their sampling rate is low, or because the volume of the 5-tuple flow is small. We propose to identify these monitors related to a 5-tuple flow with the help of routing information. Largely deployed link-state protocols like OSPF and IS-IS can provide such information. If such routing information is not available at the central unit, one has to limit the observations to monitors that have seen the flow knowing well that this might cause a bias against 5-tuple flows that got unsampled. This bias is expected to be small when aggregating over aggregate flows F_i.

According to Equation (1), we estimate the volume of a 5-tuple flow as being the sum of the weighted sum of the local estimators done using the monitors along its path. This gives the following global estimator for 5-tuple flow j belonging to aggregate flow F_i,

$$\hat{S}_{ji} = \sum_{k \in \varphi_{ji}} \lambda_k s_{kji}/p_k, \quad \text{with} \quad \lambda_k = \frac{\frac{1}{Var(\hat{S}_{kji})}}{\sum_{l \in \varphi_{ji}} \frac{1}{Var(\hat{S}_{lji})}}.$$

φ_{ji} is the set of monitors on the path followed by S_{ji}. Replacing the variances by their expressions given in the previous section, substituting the second equation into the first one, and simplifying by S_{ji}, we get,

$$\hat{S}_{ji} = \frac{1}{\alpha_{ji}} \sum_{k \in \varphi_{ji}} \beta_{kji} s_{kji},$$

with $\alpha_{ji} = \sum_{l \in \varphi_{ji}} \frac{p_l}{(1-p_l)}$ and $\beta_{kji} = \frac{1}{(1-p_k)}$. Note in particular how the α_{ji} and the β_{kji} are the same for all 5-tuple flows that follow the same path, which eases a lot the calculation. As for the variance of this estimator of 5-tuple flow sizes, it is simply equal to $Var(\hat{S}_{ji}) = S_{ji}/\alpha_{ji}$. The original flow size being unknown, we can simply substitute it by its global estimator $\hat{S}_{ji}$.

D. Reconfiguring monitors

As shown in the previous section, the variance (or mean square error) of 5-tuple flow size estimation is very important for the determination of the global system accuracy and for the identification of the monitors that should be reconfigured. For 5-tuple flow S_{ji} and monitor k we can write,

$$\frac{\partial Var(\hat{S}_{ji})}{\partial log(p_k)} = \frac{-S_{ji}.p_k}{\alpha_{ji}^2(1-p_k)^2}.$$

This represents the marginal gain in the accuracy (loss in the variance) when the logarithm of the sampling rate of monitor k is increased by a small step δ and this is from the perspective of estimating the size in packets of flow S_{ji}. As expected, this gain is positive when someone increases the sampling rate (more sampling means more accuracy). It also decreases when p_k increases, which suggests that the estimation error follows well a continuously decreasing and convex function with the sampling rate, a condition required for the uniqueness of solution in non-linear optimization theory.

By plugging the above expression in Equation (2) we obtain the utility function of the monitor k. This equation sums the accuracy and normalizes it over all 5-tuple flows forming the traffic of interest. It gives the total loss in accuracy when the sampling rate of monitor k is tuned down by a multiplicative step (additive in the logarithmic scale). By testing all monitors, we can find the best sampling rates to tune down in case of saturation. We choose to decrease the monitors having utility function values less than the average over the different monitors. Note that the sum in (2) can be calculated online as long as more reports are received. The parameters α_{ji} can be calculated only once for each configuration and for all possible paths across the network.

TABLE I
TRAFFIC TRACES SUMMARY

Trace	Start time	End time	Avg Rate	# of flows	# of pkts
S	00:30	02:30	26.34 Mbps	3250616	56178542
V	13:00	15:00	30.26 Mbps	3278041	69499589

V. EXPERIMENTAL SETUP

In order to evaluate the performance of our system, we developed an experimental platform [8]. This platform has the following main features: *(i)* it takes as input a real traffic captured on any transit link then it spreads and plays it back over an emulated network topology, *(ii)* it includes real NetFlow-like tool for traffic monitoring on all router interfaces of the emulated topology, and *(iii)* it implements the central unit as it should be in reality. In addition to validating the efficiency of the algorithms, we are particularly concerned by their feasibility and their practical deployment.

A. Emulation platform

Our experimental platform called MonLab[1], is composed of three services: *(i)* the traffic emulation service, *(ii)* the traffic monitoring and sampling service, and *(iii)* the data collection and analysis service.

Routers can be either virtual nodes connected by virtual links, or real routers connected by real links. The first service is responsible of generating the emulated traffic across the network routers. The second service implements packet sampling and flow monitoring a la NetFlow on each router interface. The later functionality is provided by SoftFlowd [9], an open source free software capable of NetFlow measurements in high speed networks. The third service mainly consists of the Flowd tool in the SoftFlowd package.

SoftFlowd requires network traffic in the TcpDump format. Unfortunately, obtaining real traffic data from an entire backbone network is a hard issue. To cope with, we proceed in the following way. We first seek unsampled packet level traces collected on high speed transit links. We consider for this

[1]MonLab: Emulation Platform for Network Wide Traffic Sampling and Monitoring, http://planete.inria.fr/MonLab/

study the ones coming from the Japanese MAWI project [10]. We parse a trace for the IP prefixes, then we dispatch them over the Autonomous Systems (ASes) connected to the edge routers of the emulated topology. The dispatching is done randomly according to some predefined weights that determine the importance of each stub AS. Furthermore, the dispatching preserves the IP prefixes in the traffic: two IP addresses belonging to the same prefix are assigned to the same AS. We leave it to the user to define the length of the prefix as a function of the granularity of the dispatching he wants to achieve. This can range from all in one AS (/0) to one IP address per AS (/32) passing by prefixes of length /16 and /24. For this work, we consider the /16 prefix as the basic unit for IP address assignment to ASes.

Once addresses are allocated, the packets in the TcpDump trace are split accordingly between the different ASes connected to the emulated topology. Shortest routes are calculated, then packets in the main TcpDump trace are associated to the different monitors over their respective paths across the network with the correct timestamps derived from the main trace. In this way, packets per monitor can be grouped into a new sub TcpDump trace and fed into SoftFlowd that can sample the packets in that monitor, form the flows and send them back to the central collector. This sampling and monitoring is done in parallel on all network router interfaces.

The different parameters of the architecture are set via an XML configuration file. The most important parameters are the topology of the studied network, the prefix length for IP address assignment and the weights of the ASes for the dispatching of packets.

B. Validation scenarios

MonLab requires the definition of a network topology over which it dispatches and replays a real traffic. This topology is supposed to connect an AS at each of its POP (Point-Of-Presence) routers. We chose to experiment over network topologies similar to the ones of well known tier-1 transit networks. Two topologies were chosen for their widely use, the Geant topology (TOPG) [11] and the Abilene one (TOPA) [12]. The weights of ASes needed for traffic dispatching are set according to the sizes of stub ASes in Geant and Abilene (we make sure these weights sum to 1). An AS of weight w will then see itself attributed $100.w\%$ of the prefixes available in the trace and will see its traffic (ingoing or outgoing) being around $100.w\%$ of the total trace traffic, both at the flow and packet levels (random prefix allocation).

Once topology and weights are set, we replay over each emulated topology different traces collected at a transpacific link by the Japanese MAWI working group [10]. Traffic traces are made by TcpDump, and then, IP addresses in the traces are scrambled by a modified version of Tcpdpriv [13]. The default scrambling configuration preserves network prefixes and IP address classes. In this paper, we present results for two traces among the many ones in this data archive: Trace S collected on 03/03/2006 during the night making the traffic relatively smooth, and Trace V collected on 03/03/2006 during the day featuring more important traffic variability. Table I provides summary information on these two traces.

We run the three services of the MonLab platform on one machine each. Machines are fast enough to follow in real time the stream of packets in the replayed TcpDump traces. There is one machine for dispatching and replaying traffic, a second machine for topology emulation and flow monitoring, and a third machine for measurement collection. This latter machine emulates the central unit; it collects NetFlow reports and implements the sampling rate adaptation algorithm.

As target application, we consider the estimation of flow sizes as described in Section IV. We recall that a flow F_i is the set of 5-tuple flows that share the same AS source and AS destination. All AS-to-AS flows are jointly considered, which is often called in the literature the traffic matrix.

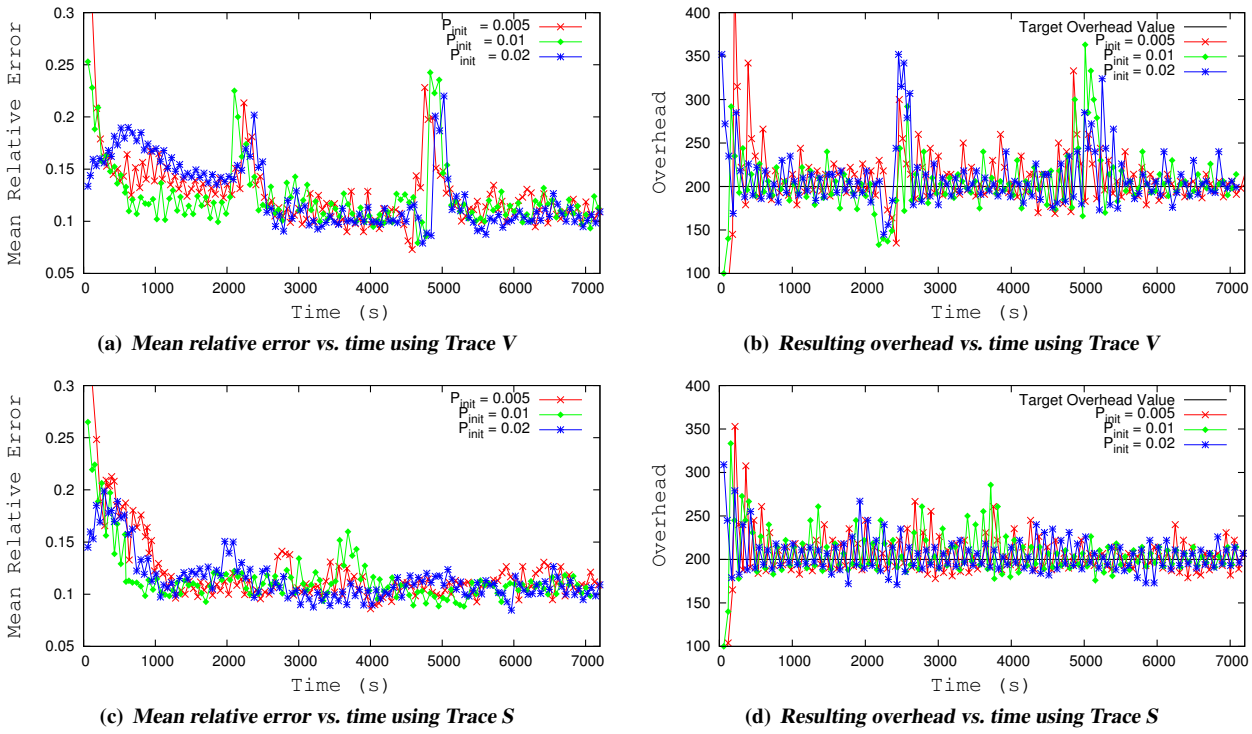
VI. Validation results

We divide this section into three parts. First, we study the efficiency and convergence of our adaptive solution and its ability to adapt to the heterogeneity of flow rates and network conditions. Second, we show the practical benefits of deploying our optimization approach by comparing it to the common static configuration approach. Last but not least, we present a global sensitivity analysis of the importance of the different parameters of our algorithm and we calculate their influence on the system behavior.

A. System efficiency, adaptability and convergence

In this section we aim to address the performance of our system using real experiments over our platform. In Figure 1, we plot the evolution of the mean relative error obtained over all AS-to-AS flows (on the left hand side) and the resulting overhead in NetFlow-records/s (on the right hand side) over time using TOPG and the two traces S and V. Each point in the graphs corresponds to an update of the sampling rates, either in the increase (t expires) or in the decrease ($\mathcal{O}$ larger than $\mathcal{TO}$). For this experiment, we set the timer t for updating sampling rates to 1 minute, the regulator σ to 2, the minimum possible sampling rate SR_{min} to 0.0005 and the maximum possible one SR_{max} to 1. The $\mathcal{TO}$ is set to 200 NetFlow-records/s.

Three initial sampling rates are considered: 0.005, 0.01 and 0.02. We can immediately observe that the system keeps improving the global accuracy while fully profiting from the available resources for measurement collection. At the beginning, the system exponentially increases sampling rates until $\mathcal{TO}$ is reached. Once done, it keeps improving the accuracy of the estimation while maintaining the overhead around its target value. After few iterations, the system reaches an equilibrium where the mean relative error tends to oscillate around its minimum value. For the smooth trace S, the equilibrium does not change much along the trace. For the other variable trace V however, we can see in the middle of the trace sudden increases in the error caused by sudden changes in the traffic. The system adapts to these changes by recalculating a new optimal configuration, always at a constant overhead. Note how the behavior is almost identical for the three initial



(a) ***Mean relative error vs. time using Trace V*** **(b)** ***Resulting overhead vs. time using Trace V***

(c) ***Mean relative error vs. time using Trace S*** **(d)** ***Resulting overhead vs. time using Trace S***

Fig. 1. The evolution of the mean relative error and the resulting overhead $\mathcal{O}$ using two different traces.

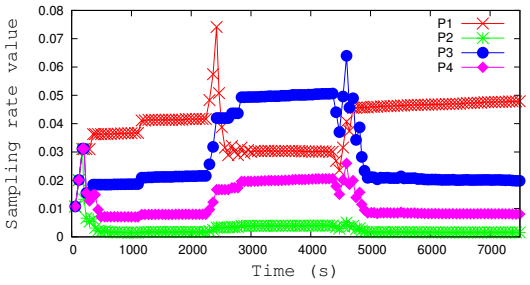


Fig. 2. Evolution of some sampling rates vs. time using trace V and TOPG.

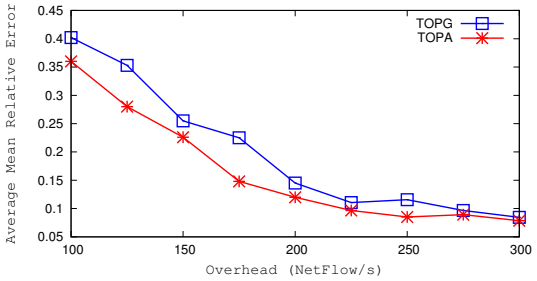


Fig. 3. Average mean relative error vs. target overhead.

sampling rates illustrating the stability of our system and its ability to converge in few iterations (few minutes here) to an equilibrium that only depends on traffic conditions and monitoring target and not on the initial configuration of sampling rates. These results are more illustrated in Figure 2, presenting the evolution of some sampling rates over time starting from an initial configuration $\mathcal{P}_{init}$ equal to 0.005 and using the trace V. We can observe the ability of our system to converge to an equilibrium in its configuration. Once done it keeps oscillating around this optimal configuration until the network conditions change. Moreover, we notice the capability of our system to track any change in the network conditions and make sampling rates move smoothly towards a new optimal configuration.

Figure 3 shows the value of the mean relative error for different $\mathcal{TO}$ values. These results are for topology TOPG and TOPA using the trace V. One can immediately notice the impact of the $\mathcal{TO}$ on the traffic estimation accuracy. For instance, using TOPG the measurement estimation error drops from 0.402 for a $\mathcal{TO}$ equal to 100 NetFlow-records/s, to 0.08 for a $\mathcal{TO}$ equal to 300 NetFlow-records/s. Indeed, for each $\mathcal{TO}$ value, the system tries to find the best configuration that minimizes the traffic estimation error. When $\mathcal{TO}$ is low, the system has to lower the sampling rates in the least significant monitors with the objective to reduce the rate of collected measurement records without much compromising the estimation accuracy. Allowing more overhead gives the system more freedom in increasing the sampling rates of the most significant monitors looking for better estimation of the sizes of the target flows. The main strength of our system is that it is able to cope with any $\mathcal{TO}$ value and provides for this value the best configuration of monitors. Now, this configuration might not satisfy the administrator in terms of the accuracy of the measurement, in this case the only remaining solution is to

increase the value of $\mathcal{TO}$. In a future research we will be working on an enhanced version of our system that adapts the $\mathcal{TO}$ in such a way to realize the measurement task with some predefined minimum accuracy. For now, we suppose the $\mathcal{TO}$ is a constraint set by the administrator and we let our system find the best configuration that maximizes accuracy given this $\mathcal{TO}$ value. The experiments over the TOPA confirm the same findings about the performance of our system. We can notice how the two curves look the same. The error for the Abilene-like topology is slightly smaller because of its smaller size and hence the larger volume of flows. Note that both experiments are conducted at equal total traffic driven by the same Trace V.

B. Comparison with the static edge method

In this section, we are interested in comparing our adaptive solution with the standard static configuration of NetFlow in order to assess the ability of our system to avoid unnecessary measurements while tracking efficiently the target flows at constant overhead. This standard solution consists of monitoring traffic at the edge of the network with static sampling rates. Each flow is monitored only one time at the input interface of the edge router of its originating AS. The problem of this solution is that it offers limited options to sample flows, and thus small flows that get mixed at their input interface with large flows suffer from a low sampling rate. Our approach has the nice feature of giving more choices for where to sample a flow, hence the protection of small flows. One has to add the dynamic feature of our approach and its ability to combine multiple measurements for the same flow and to limit the overhead. We use for the comparison two specific accounting applications:

- Traffic matrix estimation: All AS-to-AS flows are considered.
- AS traffic estimation: The focus is on the total volume of traffic generated by each stub AS.

For this experimentation, we use the Geant-like topology and the variable traffic trace V. The parameters of the experimentation are set as in the previous sections. For the sampling rate in the case of the static edge configuration, denoted by p, we set it in such a way that the resulting reporting overhead is the same as in our network-wide adaptive case, and this is for the main purpose of fairness between the two approaches. If N_S is the total number of 5-tuple flows in the trace, D the duration of the trace, $\pi(S)$ the probability to sample a 5-tuple flow of size S packets, S being a random variable, then the sampling rate p is set to satisfy the following:

$$\frac{N_S.\pi(S)}{D} = \frac{N_S.E[1-(1-p)^S]}{D} = \mathcal{TO}.$$

The term on the right-hand side is no other than the target overhead of our adaptive architecture. The term on the left-hand side is an estimation of the rate of collected records in the static edge configuration.

Traffic matrix estimation: While giving on average close performance to our approach, the edge solution presents sampling bias against small flows as shown in Figure 4 for the smallest 20 flows. This figure plots the average mean relative error as a function of $\mathcal{TO}$. With the edge solution, small flows dilute within large flows and suffer from low estimation accuracy. If this happens, no other choices are available to sample them elsewhere. However, with our approach, we are able to track small flows on other lightly loaded links inside the network and combine measurements from different routers together without incurring more overhead on the system. As we can see, in order to track small flows using the edge solution with a similar accuracy to the one we obtain using the adaptive solution, we have to use a $\mathcal{TO}$ value larger than 150% of the value used by the adaptive solution.

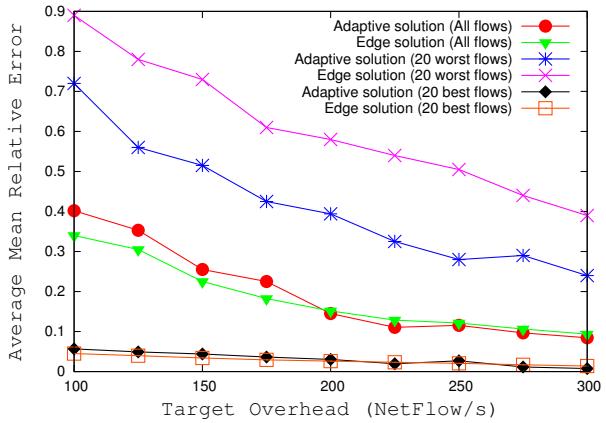


Fig. 4. The average mean relative error of flow measurements: Our approach vs. static edge one.

AS traffic estimation: We change our objective and instead of defining a flow as being the volume of traffic from one stub AS to another stub AS, we define it as the total volume of traffic generated by each stub AS. We count both the outgoing and ingoing traffic for each AS. The best configuration is the one that minimizes the sum of mean square relative errors of AS traffic estimators. Changing target measurement is very easy in the context of our approach; one has to correctly define an aggregate flow. In this scenario, the traffic volume of an AS should be proportional to the weight attributed to it during the trace dispatching phase.

To further prove the generality and efficiency of our approach in compared to the standard static edge one, we perform two tasks within this scenario. In a first time, we estimate the traffic volume of the different ASes (All ASes task). Then, we estimate the traffic volume of ASes contributing to more than some percentage of the total traffic trace (Large ASes task). We present results for a 6% threshold. Some ASes are smaller than this threshold but there traffic might still be reported to the central collector, yet they are not included in the optimization loop and are not returned to the monitoring application. The main purpose of our architecture is to reduce the volume of undesirable traffic. The All ASes task is a particular case of this second general task and can be obtained by setting the threshold to 0%. Table II presents a summary

TABLE II
COMPARING AS TRAFFIC VOLUME ESTIMATIONS.

AS	Affected weight	Without sampling		Edge sloution		All ASes		Large ASes	
		AS ratio	# of pkts	AS ratio	# of pkts	AS ratio	# of pkts	AS ratio	# of pkts
2	10	9.926	13797278	8.91	14273284	10.88	12937707	9.35	14040661
5	8	6.66	9258345	5.31	8499160	7.08	8415835	6.568	9860137
4	7	8.13	11302633	6.616	10590567	8.74	10387119	7.928	11901672
7	6	7.23	10049975	5.782	9256026	7.704	9155527	7.1	10663023
8	5	6.03	8381660	4.749	7602165	6.376	7577020	5.968	8959994
11	4	3.04	4228675	1.807	2892413	2.975	3535172	0.45	676588
9	2	1.87	2599884	0.633	1013954	1.542	1832918	0.155	233989
Total number of packets		138999178		160057553		118830397		150119112	

of the experimental results for a selection of ASes. The first two columns present the AS number and its associated weight. The other columns present the AS traffic volume estimation in number of packets and the ratio of this volume with respect to the total estimated network traffic. Four configurations are presented, the one without sampling as a reference configuration, the static edge one, the adaptive All ASes one, and the adaptive Large ASes one. A set of observations can be made from these results. The first observation is that the All ASes adaptive configuration provides more accurate results for all AS traffic volumes independently of their sizes. The traffic volume of ASes is better estimated than with the static edge configuration, especially for small ASes whose traffic get diluted within the traffic of large ASes if only sampled at the edge. The second observation we can make is that for large ASes, one can even get a better estimation by only focusing in the optimization on the sizes of these large ASes. As requested, small ASes contributing to less than 6% of the total network traffic get ignored by our optimization, hence the decrease in their accuracy. Indeed, the overhead these small ASes generate with the All ASes configuration and the static edge one is used to better sample the large ASes and to better estimate their traffic volumes. These results illustrate the adaptive nature of our approach and its capacity to cope with the monitoring application needs, always at constant monitoring overhead.

C. Global sensitivity analysis

In the previous two parts, we gave a particular attention to the impact of the $\mathcal{TO}$. Yet, the system has other parameters and it is important to evaluate their impact as well. In this part, we demonstrate indeed that, apart from the $\mathcal{TO}$, the other parameters have minor impact on system performance.

The goal of global sensitivity analysis is to characterize, qualitatively or quantitatively, what impact an input parameter has on a system output and how it compares with the impact of the other parameters. Fourier Amplitude Sensitivity Test(FAST) [14], [15] is one of the most efficient methods in sensitivity analysis [16], [17]. Among its advantages are: fast implementation, possibility to deal with nonmonotonic models, arbitrary large variations in input parameters, and no need for the knowledge of the mathematical model.

The main idea of FAST is to assign to each parameter a distinct integer frequency (characteristic frequency). Then, for a specific parameter, the variance contribution can be singled out of the model output with the help of the Fourier transformation. Therefore, FAST is also referred to as variance based sensitivity analysis. Specifically, let us consider a nonlinear model $y = f(x_1, x_2, ..., x_n)$ where x_n are parameters. We emphasize that the FAST method does not require the analytic knowledge of the function $f(\cdot)$. Various search functions have been proposed. The search function must let the parameter x_i to oscillate with frequency ω_i. For instance, the authors of [18] have proposed the search function $x_i = \frac{1}{2} + \frac{1}{\pi}\arcsin(\sin(\omega_i s))$, which is a particular case of a more general search function [19]

$$x_i = F_i^{-1}\left(\frac{1}{2} + \frac{1}{\pi}\arcsin(\sin(\omega_i s))\right), \tag{3}$$

where $F_i^{-1}(\cdot)$ is the inverse cumulative distribution function for x_i. To make more efficient use of the model evaluations, the authors of [18] have suggested the following modification

$$x_i = \frac{1}{2} + \frac{1}{\pi}\arcsin(\sin(\omega_i s + \varphi_i)), \tag{4}$$

where φ_i is a random phase-shift chosen uniformly in the interval $[0, 2\pi]$. The model output becomes a periodic function with period 2π. Thus, we can represent the model with a Fourier series,

$$y = f(x_1, x_2, ..., x_n) = A_0 + \sum_{k=1}^{\infty}[A_k\cos(ks) + B_k\sin(ks)].$$

If we denote a sample of size N as $S = \{s_1, s_2, ..., s_N\}$, then, using either (3) or (4) as a search function, we can obtain the sampled values of the parameters $X_i = \{x_{i1}, x_{i2}, ..., x_{iN}\}$, and the discrete Fourier transform coefficients $A_0 = \frac{1}{N}\sum_{j=1}^{N} f(s_j)$,

$$A_k = \frac{2}{N}\sum_{j=1}^{N} f(s_j)\cos(s_j k) \text{ and } B_k = \frac{2}{N}\sum_{j=1}^{N} f(s_j)\sin(s_j k),$$

where $f(s_j) = f(x_{1j}, ..., x_{nj})$ and $k = 1, ..., (N-1)/2$.

The variance of the model output can be decomposed into variance components at the integer frequencies,

$$V = \frac{1}{2}\sum_{k=1}^{(N-1)/2}[A_k^2 + B_k^2],$$

By summing the spectrum values $\Lambda_k = [A_k^2 + B_k^2]/2$ for the characteristic frequencies ω_i and their higher harmonics, the

TABLE III
PARAMETERS OF THE EXPERIMENT.

Parameter	symbol	range	impact
Target Overhead	$\mathcal{TO}$	$[20, 500]$	0.58
Increasing sampling rates timer	t	$[60s, 300s]$	0.0142
γ regulator	σ	$[1, 10]$	0.00747
The value of $\mathcal{P}_{init}$	p_{init}	$[0.005, 0.02]$	0.01179
Minimum sampling rate value	SR_{min}	$[0.0005, 0.005]$	0.00691
Maximum sampling rate value	SR_{max}	$[0.02, 1]$	0.00721

partial variance in model output arising from the uncertainty of parameter x_i, V_i, can be estimated by $V_i = \sum_p \Lambda_p \omega_i$, where $p\omega_i \leq (N-1)/2$. The ratio V_i/V measures the contribution of parameter x_i. This ratio is also referred to as the first-order sensitivity index [20].

Because the characteristic frequencies are integers, there will be an aliasing effect if one frequency is a linear combination of the others. It is said that a frequency set is free of interferences to an order M if

$$\sum_{i=1}^{n} a_i \omega_i \neq 0, \quad \sum_{i=1}^{n} |a_i| \leq M + 1,$$

where a_i is an integer and M is a design integer (usually 4 or 6). In order to avoid the interference effect the maximal value of p in calculating V_i should be M. In [15] the authors have proposed the following empirical formula for calculating the characteristic frequencies free of interference up to order $M = 4$: $\omega_1 = \Omega_n$, and $\omega_i = \omega_{i-1} + d_{n+1-i}, \quad i = 2, ..., n.$. The parameters Ω_n and d_k can be found in a table provided in [15]. Below we give several line from that table.

Dimension, n	Ω_n	d_n	Minimal number of points, N_{min}
1		4	
2		8	
3	1	6	38
4	5	10	78
5	11	20	142
6	1	22	182

For instance, for the case of six input parameters we obtain the following values of the characteristic frequencies $\omega_1 = \Omega_6 = 1$; $\omega_2 = \omega_1 + d_5 = 21$; $\omega_3 = \omega_2 + d_4 = 31$; $\omega_4 = \omega_3 + d_3 = 37$; $\omega_5 = \omega_4 + d_2 = 45$; $\omega_6 = \omega_5 + d_1 = 49$; We have applied the method FAST to our system to characterize the impact of the different parameters used in experimentations on results. Table III summarizes the different evaluated parameters with their ranges. The last column presents the impact of each parameter on the system output.

It is immediately noticed that the parameter having most important impact on the system output is the target overhead $\mathcal{TO}$ while the other parameters have a light impact on results in the order of 1% or less. Indeed, for some value of $\mathcal{TO}$, there is an optimal configuration of monitors, and our system will converge to this optimal configuration in a robust manner with respect to the other parameters. It is only by changing the value of $\mathcal{TO}$ that the system will converge to another optimal configuration yielding another measurement precision.

VII. CONCLUSIONS

We presented a network-wide monitoring system that adopts an adaptive centralized approach to coordinate responsibilities across monitors in order to address the tradeoff between monitoring accuracy and overhead. We have also developed MonLab, an experimental platform to evaluate the performance of our system. Experimental results proved the ability of our system to continuously improve the monitoring accuracy while limiting the overhead to its target value $\mathcal{TO}$. Compared to static edge configuration, our system has shown its advantages in better capturing network flows especially for small flows. Moreover, we provided a global sensitivity study of the impact of the different parameters on the system behavior.

Our future research will focus on the validation of our approach with more applications. Candidate applications are the counting of flows, the tracking of some user-specific flows and the detection of anomalies. The distribution of the control and the adaptive tuning of the overhead are other interesting objectives to realize.

REFERENCES

[1] N. Duffield, C. Lund, and M. Thorup, "Learn more, sample less: control of volume and variance in network measurement," *IEEE Transactions in Information Theory*, vol. 51, pp. 1756–1775, 2005.

[2] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," in *Proc. of ACM SIGCOMM*, 2004.

[3] N. Duffield, C. Lund, and M. Thorup, "Properties and prediction of flow statistics from sampled packet streams," in *Proc. of IMC*, 2002.

[4] V. Sekar, M. Reiter, W. Willinger, H. Zhang, R. Kompella, and D. Andersen, "cSamp: A system for network-wide flow monitoring," in *Proc. 5th USENIX NSDI*, 2008.

[5] G. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran, "Reformulating the monitor placement problem: Optimal networkwide sampling," in *Proc. of CoNeXT*, 2006.

[6] Cisco, "Netflow services and applications," *White Paper*, 2000.

[7] N. Duffield, C. Lund, and M. Thorup, "Optimal combination of sampled network measurements," in *IMC 2005*, 2005.

[8] A. Krifa, I. Lassoued, and C. Barakat, "Emulation platform for network wide traffic sampling and monitoring," *TRAC*, 2010.

[9] , "Softflowd flow-based network traffic analyser," http://www.mindrot.org/projects/softflowd/.

[10] "Mawi working group traffic archive," http://tracer.csl.sony.co.jp/mawi/.

[11] , "Geant: The european research and academic backbone." http://www.geant.net/.

[12] "Abilene or internet2: The us research and acamdemic backbone." http://www.internet2.edu/network/.

[13] "Tcpdpriv: A program for eliminating confidential information from packets." http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html/.

[14] R. Cukier, C. Fortuin, K. Shuler, A. Petshek, and J. Schaibly, "Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory," *The Journal of Chemical Physics*, 1973.

[15] R. Cukier, J. Schaibly, and K. Shuler, "Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. iii analysis of the approximations," *The Journal of Chemical Physics*, vol. 63, 1975.

[16] A. Saltelli, K. Chan, and M. Scott, "Sensitivity analysis," *Wiley*, 2000.

[17] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, "Sensitivity analysis in practice: A guide to assessing scientific models," *Wiley*, 2004.

[18] A. Saltelli, S. Tarantola, and K. P.-S. Chan, "A quantitative model-independent method for global sensitivity analysis of model output," *Technometrics*, vol. 41, pp. 39–56, 1999.

[19] Y. Lu and S. Mohanty, "Sensitivity analysis of a complex, proposed geologic waste disposal system using the fourier amplitude sensitivity test method," *Reliab. Eng. syst. Safe.*, vol. 72, 2001.

[20] I. Sobol, "Sensitivity estimates for nonlinear mathematical models," *Math. Model. Comput. Exp.*, vol. 1, pp. 407–414, 1993.